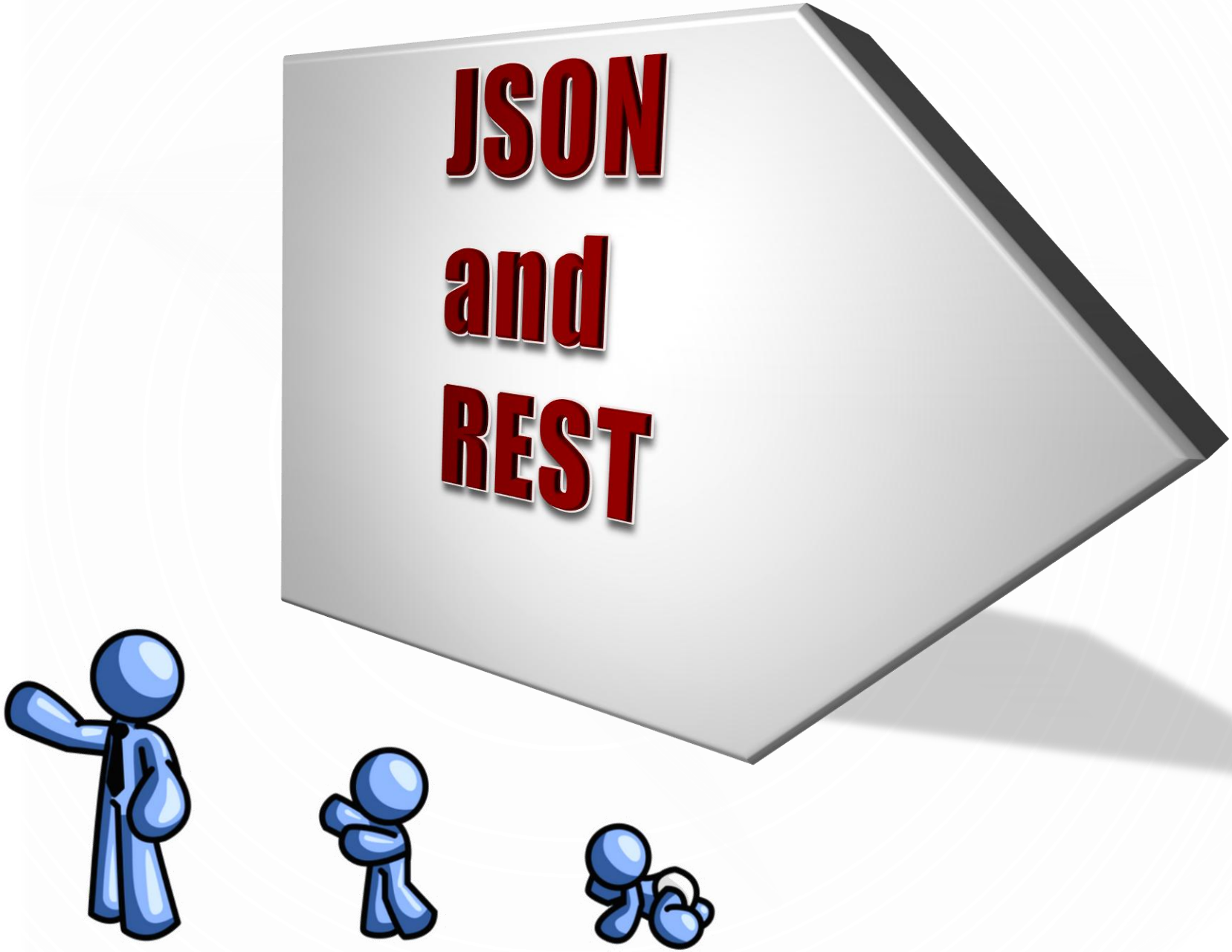
Based on http://www.slideshare.net/rmaclean/json-and-rest
and http://www.slideshare.net/hburakcetinkaya/rest-res-tful-web-services



In a Nutshell

REST is about resources and how to represent resources in different ways.

REST is about client-server communication.

REST is about how to manipulate resources.

REST offers a simple, interoperable and flexible way of writing web services that can be very different from other techniques.

Comes from Roy Fielding's Thesis study.

REST

Representational **S**tate **T**ransfer

Architectural style (technically not a standard)

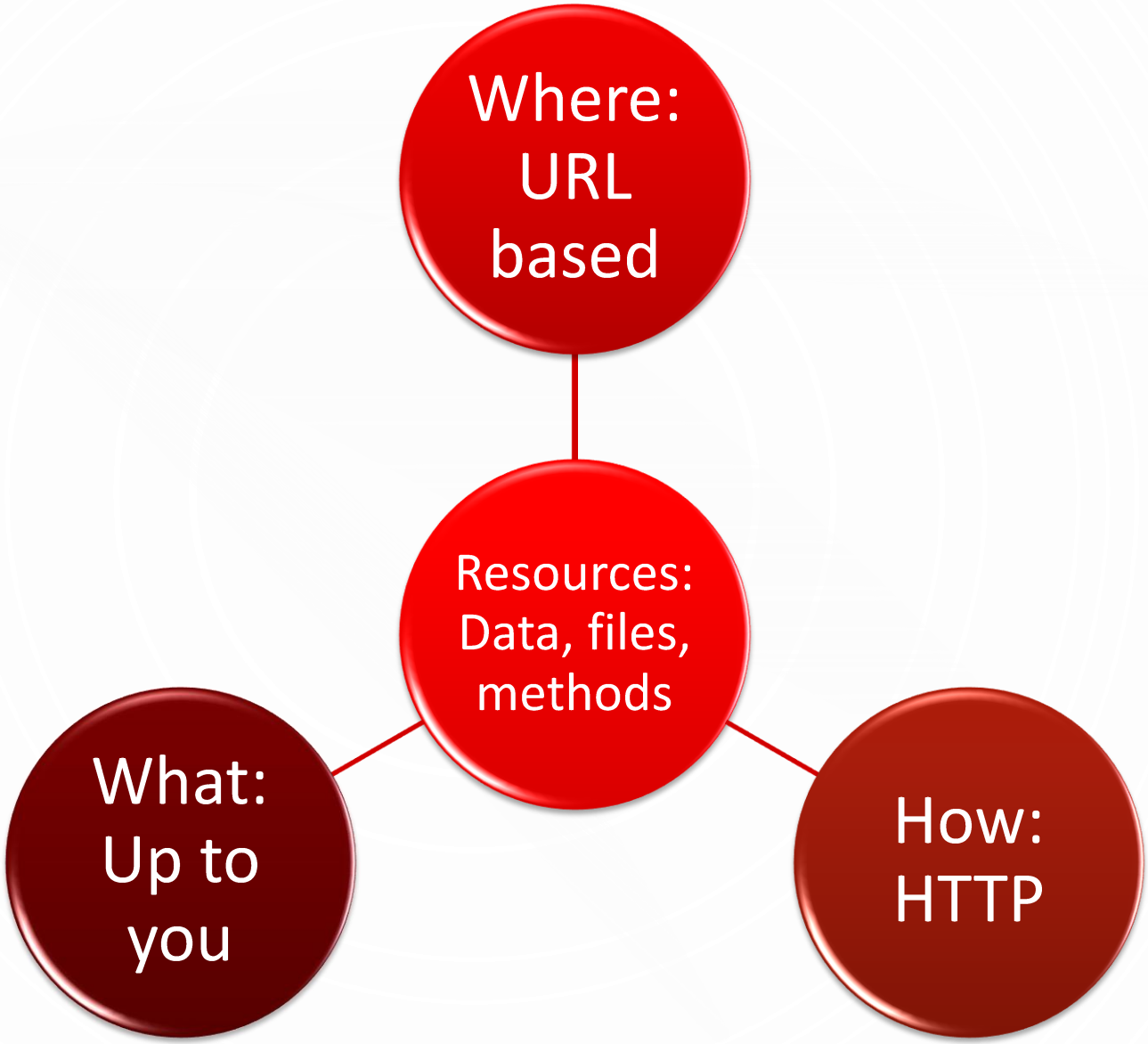
Idea: a network of web pages where the client progresses through an application by selecting links

When client traverses link, accesses new resource (i.e., transfers state)

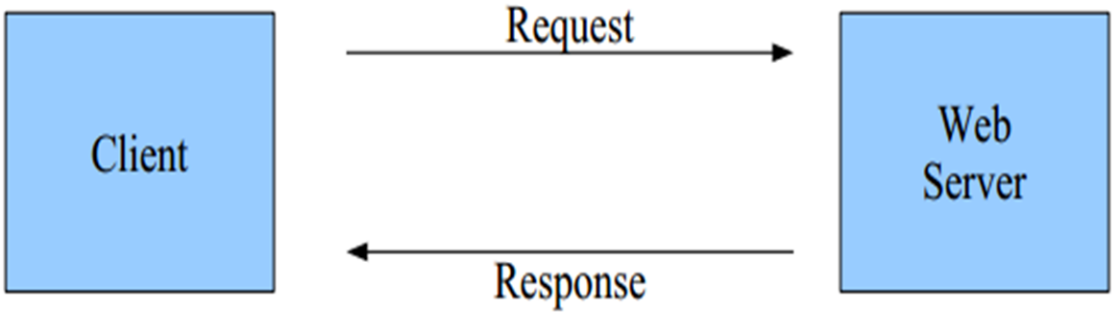
Uses existing standards, e.g., HTTP

REST is an architecture all about the Client-Server communication.

REST – Core Principal



THE WEB



REST

Client **request**s a specific **resource** from the server.

The server **respond**s to that request by delivering the requested resource.

Server does not have any information about any client.

So, there is no difference between the two requests of the same client.

A model which the representations of the resources are transferred between the client and the server.

The Web as we know is already in this form!

Resources

Resources are just consistent mappings from an identifier [such as a URL path] to some set of views on server-side state.

Every resource must be uniquely addressable via a URI.

“If one view doesn’t suit your needs, then feel free to create a different resource that provides a better view. ”

“These views need not have anything to do with how the information is stored on the server … They just need to be understandable (and actionable) by the recipient.” *Roy T. Fielding*

REST – Where/How: Simple Example

Premise:

- Data in a table could be a resource we want to read

Database server called *bbddb01*

Database called *northwind*

Table called *users*

http://bbddb01/northwind/users

What, What, What?

What type of content you return is up to you.

Compare to SOAP where you must return XML.

Most common are XML or JSON. You could return complex types like a picture.

REST – Is it used?

Web sites are RESTful

RSS is RESTful

Twitter, Flickr and Amazon expose data using REST

Some things are “accidentally RESTful” in that they offer limited support.

Real Life: Flickr API

Resource: Photos

Where:

- http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}.jpg
- http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}_[mstb].jpg
- http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{o-secret}_o.(jpg|gif|png)

What: JPEG, GIF or PNG (defined in the URL)

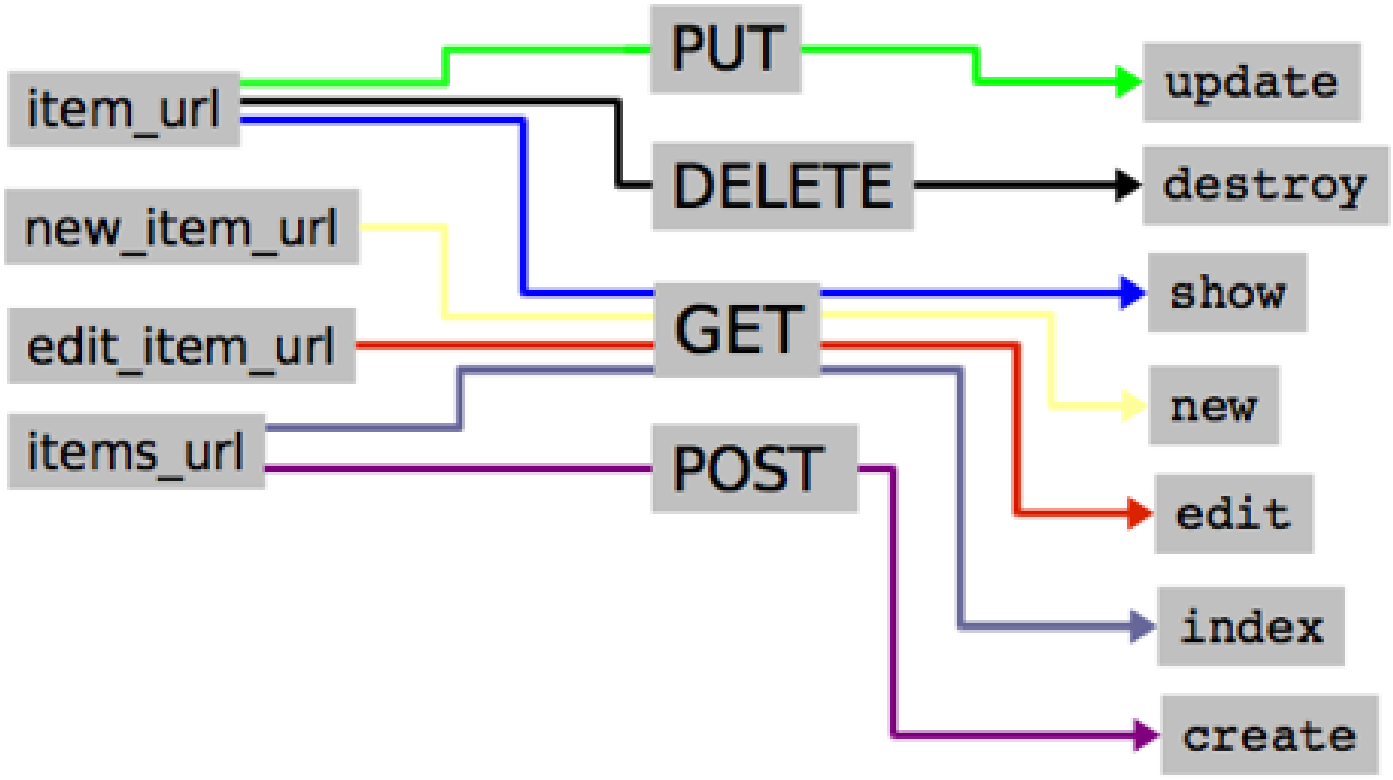
http://farm1.static.flickr.com/2/1418878_1e92283336_m.jpg

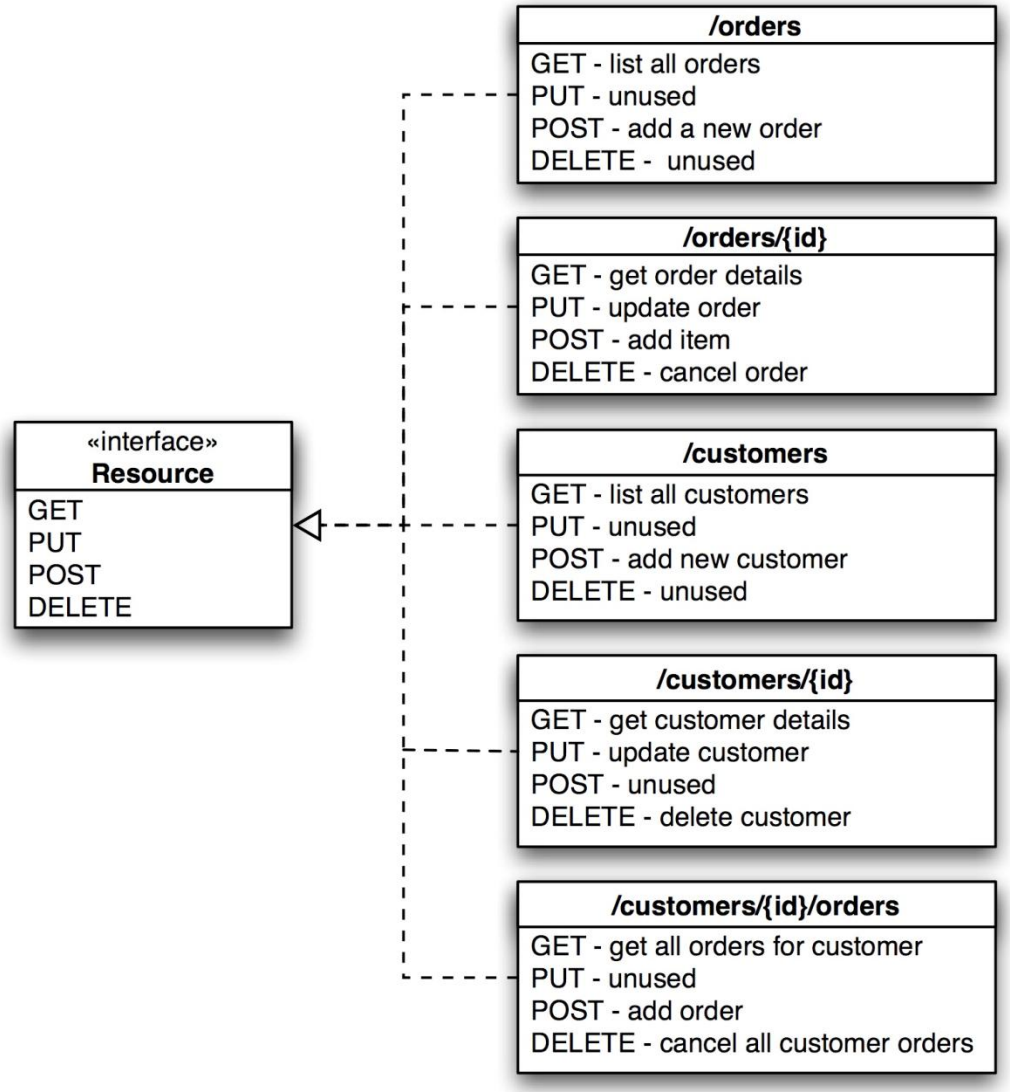
REST – Methods

HTTP Methods are a key corner stone in REST.
They define the action to be taken with a URL.
Proper RESTful services expose all four – "accidental" expose less.
Nothing stopping you doing some Mix & Match
❖Some URL's offering all of them and others a limited set

What are the four methods and what should they do?

REST	CRUD (Create, Read, Update, Delete)
POST	Create
GET	Read
PUT	Update or Create
DELETE	Delete


item_url
new_item_url
edit_item_url
items_url
PUT
DELETE
GET
POST
update
destroy
show
new
edit
index
create


«interface»
Resource
GET
PUT
POST
DELETE
/orders
GET - list all orders
PUT - unused
POST - add a new order
DELETE - unused
/orders/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order
/customers
GET - list all customers
PUT - unused
POST - add new customer
DELETE - unused
/customers/{id}
GET - get customer details
PUT - update customer
POST - unused
DELETE - delete customer
/customers/{id}/orders
GET - get all orders for customer
PUT - unused
POST - add order
DELETE - cancel all customer orders

REST – Methods Example

http://bbddb01/northwind/users[firstname="rob%"]

- **+ POST = Error**
- **+ GET = Returns everyone who begins with rob**
- **+ PUT = Error**
- **+ DELETE = Deletes everyone who begins with rob**

http://bbddb01/northwind/users

& we add some input data

- **+ POST = Creates a new user**
- **+ GET = Returns everyone who meets criteria**
- **+ PUT = Creates/Updates a user (based on data)**
- **+ DELETE = Deletes everyone who meets criteria**

REST – Methods Example

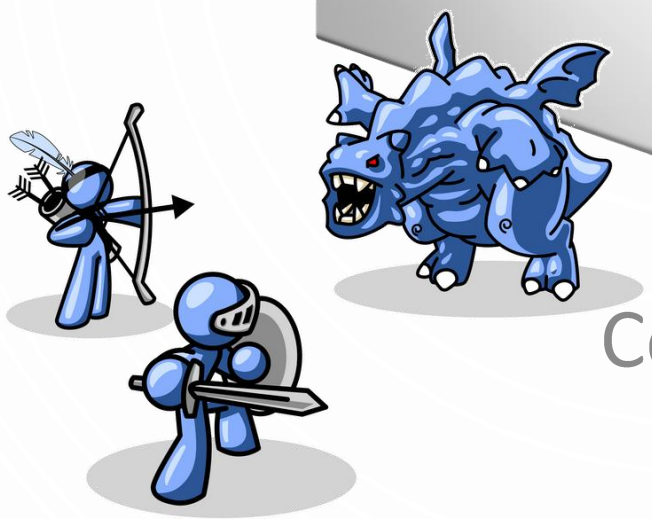
http://bbddb01/northwind/users[firstname="rob%"]

- **+ POST = Error**
- **+ PUT = Error**

What would the error be?

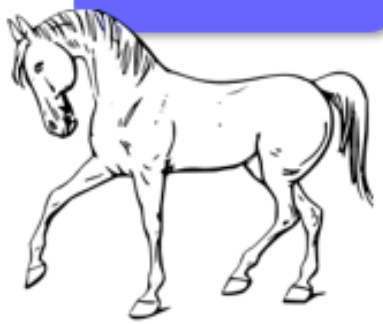
HTTP 400 or 500 errors are normally used to indicate problems – same as websites

FIGHT:
REST vs. SOAP



Comparing apples and oranges

HTTP



REST

Rides directly on HTTP. Plain and simple. In reality, this is all you need to send data from point A to point B and get the required response. Catch: Until something that represents a service contract is put in to place, it's kinda "anything goes".

SOAP

The coach is your SOAP envelope: it wraps your data. Main strength is the presence of a contract: the WSDL. Gives you the "comfort" of easily generating artifacts. Catch: look at the complexity and added weight.

Consider the cutie on the left as your data

SOAP

Client  +  =  <- -> Server

Data + SOAP Standards

Your data sent to the server would become as huge as the chubby above

REST

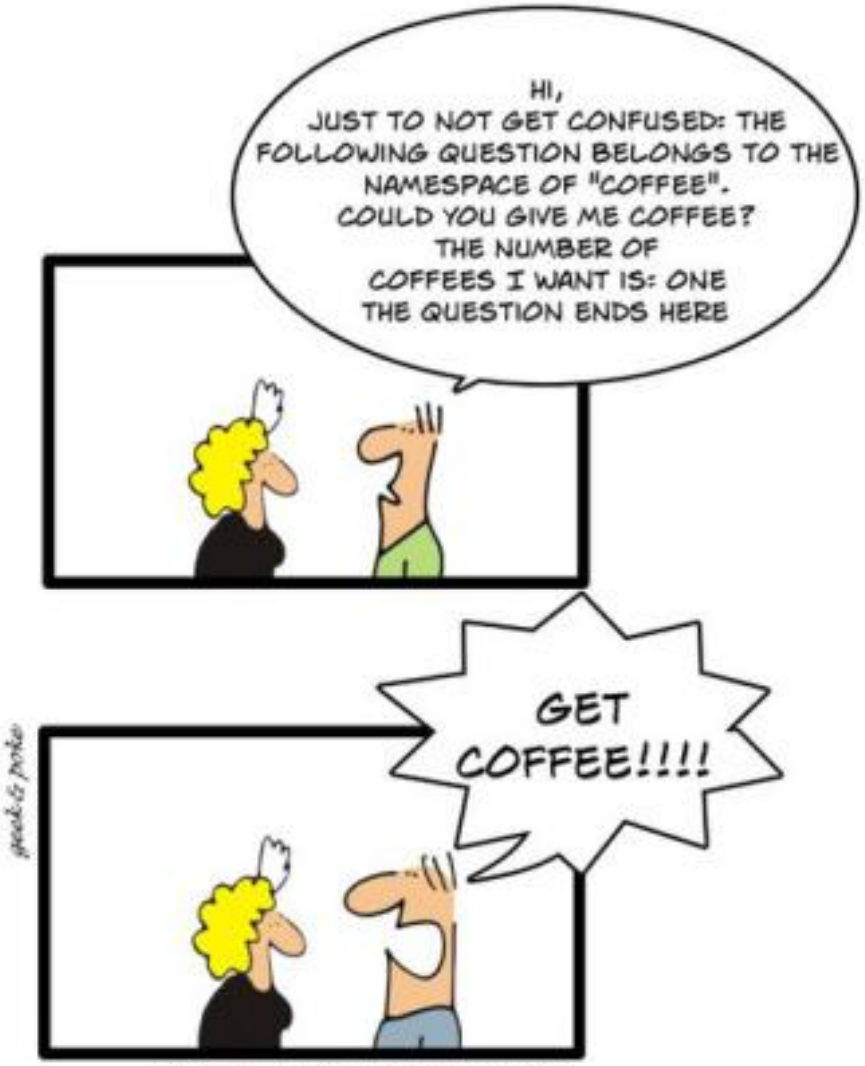
Client 

<- ->

Rest is like sending the DATA as such

Server

SERVICE CALLING MADE EASY



RULE 1: SOAP IS MUCH MORE POLITE THAN REST

JSON Basics

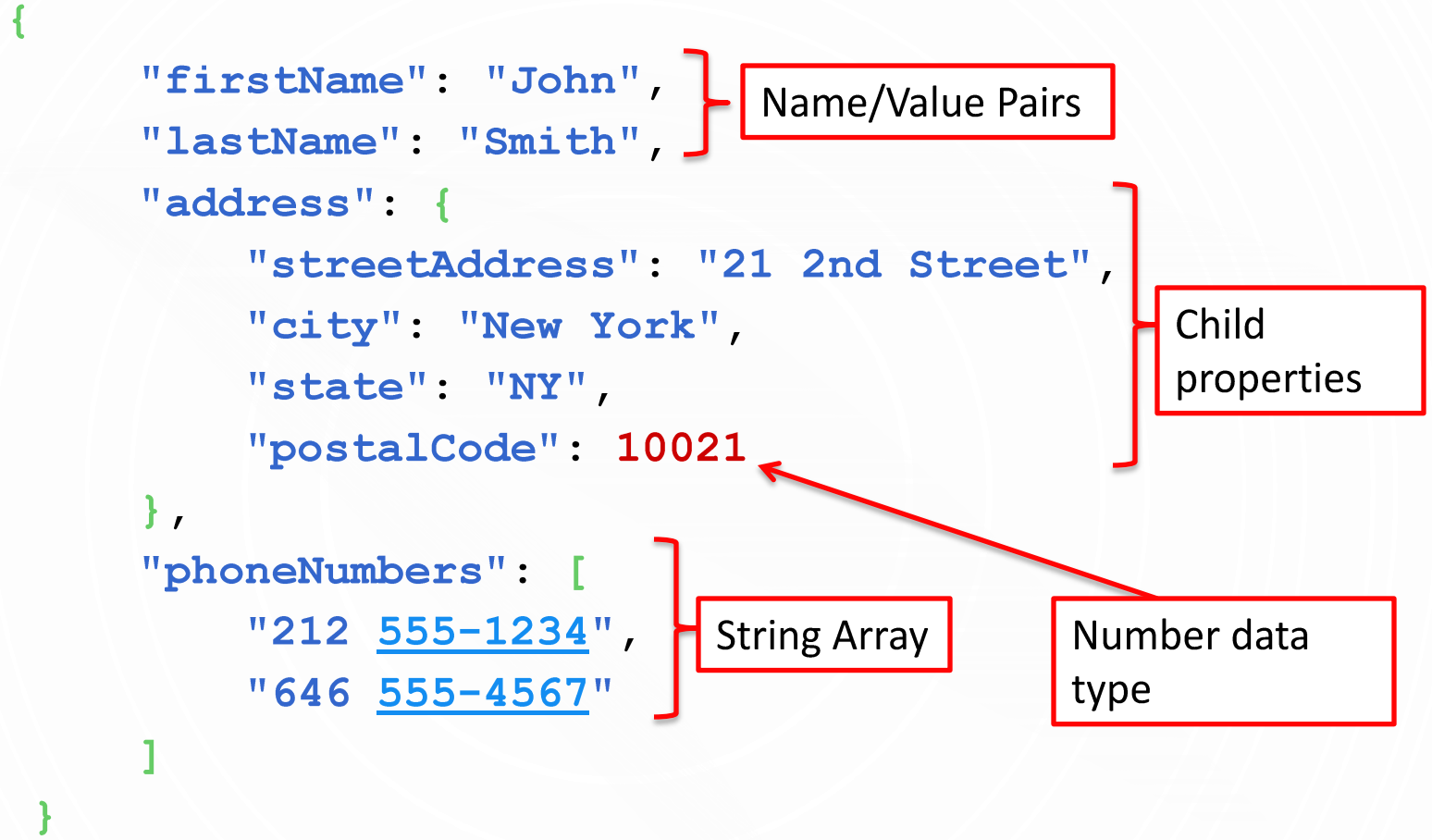
Demo

JSON – What is it?

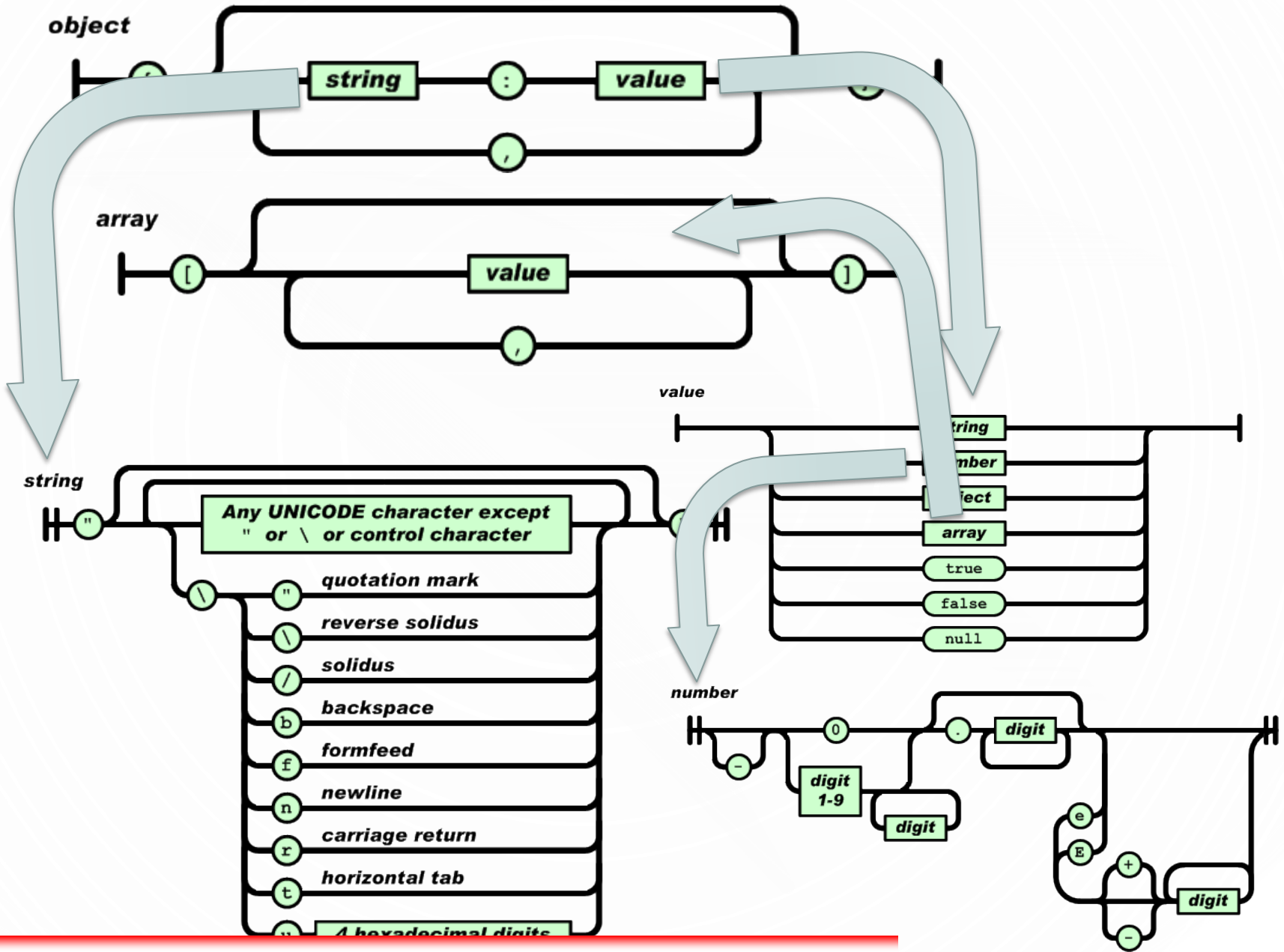
*"JSON (JavaScript Object Notation) is a **lightweight data-interchange format**. It is easy for humans to read and write. It is easy for machines to parse and generate"* – JSON.org

Importantly: JSON is a subset of JavaScript

JSON – What does it look like?



JSON Data Structures



Aren’t they the same?

#	XML	JSON
1	XSD Validation	No Validation
2	Includes namespaces	No namespaces
3	Requires parsing. xml doc is parsed through parameters like xpath etc.	Parsing is just an eval. Therefore it is fast.
4	JavaScript work with strings. May require additional parsing	In JavaScript, we can work with objects-run time evaluation of types
5	Write JavaScript to parse the xml objects	A JavaScript object
6	Heavy compared to JSON	JSON is light weight compared to XML

```
<empinfo>
  <employees>
    <employee>
      <name>Scott Philip</name>
      <salary>£44k</salary>
      <age>27</age>
    </employee>
    <employee>
      <name>Tim Henn</name>
      <salary>£40k</salary>
      <age>27</age>
    </employee>
    <employee>
      <name>Long yong</name>
      <salary>£40k</salary>
      <age>28</age>
    </employee>
  </employees>
</empinfo>
```

```
{ "empinfo" :
    {
        "employees" : [
        {
          "name" : "Scott Philip",
          "salary" : £44k,
          "age" : 27,

        },

        {
          "name" : "Tim Henn",
          "salary" : £40k,
          "age" : 27,
        },

        {
         "name" : "Long Yong",
         "salary" : £40k,
         "age" : 28,

        }
                    ]
    }
}
```

JSON vs. XML

Which of them should you use?

Use Both – They both have strengths and weaknesses and you need to identify when one is stronger than the other.